Annals of Pure and Applied Logic 163 (2012) 854–874

Contents lists available at SciVerse ScienceDirect

Annals of Pure and Applied Logic

journal homepage: www.elsevier.com/locate/apal

Light linear logics with controlled weakening: Expressibility, confluent strong normalization

Max Kanovich

Queen Mary University of London, Computer Science Department, Mile End Road, London E1 4NS, UK

ARTICLE INFO

Article history:
Available online 26 October 2011

MSC:
03B47
03B70
03D15
68Q17

Keywords:
Light linear logic
Elementary linear logic
Proofs-as-programs paradigm
Cut-elimination-as-computation paradigm
Implicit computational complexity
Polynomial-time strong normalization

ABSTRACT

Starting from Girard's seminal paper on light linear logic (LLL), a number of works investigated systems derived from linear logic to capture polynomial time computation within the *computation-as-cut-elimination* paradigm.

The original syntax of LLL is too complicated, mainly because one has to deal with sequents which do not just consist of formulas but also of 'blocks' of formulas.

We circumvent the complications of 'blocks' by introducing a new modality ∇ which is exclusively in charge of 'additive blocks'. One of the most interesting features of this purely multiplicative ∇ is the possibility of the second-order encodings of additive connectives.

The resulting system (with the traditional syntax), called Easy-LLL, is still powerful to represent any deterministic polynomial time computations in purely logical terms.

Unlike the original LLL, Easy-LLL admits polynomial time strong normalization together with the Church–Rosser property, namely, cut elimination terminates in a *unique way* in polytime by any choice of cut reduction strategies.

1. Introduction, motivation, summary

Girard [8] has introduced light linear logic (LLL) as a refinement of the 'proofs-as-programs' paradigm to polynomial-time computation.

The basic idea of the representability of computation in *purely logical terms* is the following.

Let every number m be associated with a certain cut-free proof π_m of a fixed formula, say $\mathtt{Num}$.

A function f mapping from non-negative integers into non-negative integers is *represented by a proof* $\widetilde{\Pi}$ of the sequent:

$$\mathtt{Num} \vdash \mathtt{Num},$$

if for any input m, a cut elimination procedure applied to the proof $\widetilde{\Pi}$ and the proof π_m, associated with the m, leads to a cut-free proof $\pi_{f(m)}$ associated with the output $f(m)$:

$$\dfrac{\dfrac{\pi_m}{\vdash \mathtt{Num}} \qquad \dfrac{\widetilde{\Pi}}{\mathtt{Num} \vdash \mathtt{Num}}}{\vdash \mathtt{Num}}(\mathrm{cut}) \quad \longrightarrow \longrightarrow \ldots \longrightarrow \quad \dfrac{\pi_{f(m)}}{\vdash \mathtt{Num}}$$

An implicit *fairness condition* is that the number of cut reductions must be in 'polynomial' accordance with the time complexity of f.

In interest of a wider audience, we illustrate this concept with the following example, in which we show how to encode *projections* and *conditional* constructs within the 'cut-elimination-as-computation' paradigm.

E-mail address: mik@dcs.qmul.ac.uk.

0168-0072/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.apal.2011.09.012

Example 1.1 (*cf. Girard [8]*)**.** Suppose that a function f is defined as:

$$f(\beta) = \text{if } \beta \text{ then } g_1 \text{ else } g_2$$

where g_1 and g_2 are represented by proofs of the form: $\dfrac{\pi_1}{\Gamma \vdash \Phi}$ and $\dfrac{\pi_2}{\Gamma \vdash \Phi}$, respectively.

Let the Boolean inputs **true** and **false** be encoded by the following proofs "**true**" and "**false**", respectively[1]:

$$\text{"}\mathbf{true}\text{"}\colon \left\{ \frac{\vdash 1}{\vdash (1 \oplus 1)}\,\mathbf{plus}_1 \qquad \text{and} \qquad \text{"}\mathbf{false}\text{"}\colon \left\{ \frac{\vdash 1}{\vdash (1 \oplus 1)}\,\mathbf{plus}_2 \right. \right. \tag{1}$$

We will encode our branching f by a proof of the form:

$$\frac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi} \qquad \dfrac{\dfrac{\pi_2}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi}}{(1 \oplus 1),\ \Gamma \vdash \Phi} \tag{2}$$

(1) **Case 1:** β is **true**.

By applying 'cut' to the proofs: "**true**" and (2), this will result in the *first projection* "π_1".

Namely, the corresponding cut:

$$\frac{\dfrac{\vdash 1}{\vdash (1 \oplus 1)}\,\mathbf{plus}_1 \qquad \dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi} \qquad \dfrac{\dfrac{\pi_2}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi}}{(1 \oplus 1),\ \Gamma \vdash \Phi}}{\Gamma \vdash \Phi}\,\mathbf{cut}$$

is reduced to the proof representing g_1:

$$\longrightarrow \quad \frac{\vdash 1 \qquad \dfrac{\dfrac{\pi_1}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi}}{\Gamma \vdash \Phi}\,\mathbf{cut} \quad \longrightarrow \quad \frac{\pi_1}{\Gamma \vdash \Phi}$$

(2) **Case 2:** β is **false**.

Similarly, by applying 'cut' to "**false**" and (2), this results in the *second projection* "π_2".

Namely, the corresponding cut:

$$\frac{\dfrac{\vdash 1}{\vdash (1 \oplus 1)}\,\mathbf{plus}_2 \qquad \dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi} \qquad \dfrac{\dfrac{\pi_2}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi}}{(1 \oplus 1),\ \Gamma \vdash \Phi}}{\Gamma \vdash \Phi}\,\mathbf{cut}$$

is reduced to the proof representing g_2:

$$\longrightarrow \quad \frac{\vdash 1 \qquad \dfrac{\dfrac{\pi_2}{\Gamma \vdash \Phi}}{1,\ \Gamma \vdash \Phi}}{\Gamma \vdash \Phi}\,\mathbf{cut} \quad \longrightarrow \quad \frac{\pi_2}{\Gamma \vdash \Phi}$$

Bringing both cases together, we conclude that (2) is a correct representation of f in Example 1.1. □

1.1. Motivations

Starting from Girard's seminal paper [8] on light linear logic, a number of works investigated systems derived from linear logic and corresponding to computational complexity classes. The intuitionistic fragments of a number of systems are shown to capture polynomial time computation within the computation-as-cut-elimination paradigm (see, for instance, [1–17]).

However, the underlying light logics do not behave well with respect to any choice of cut reduction strategies.

Thus the original LLL has been proved to be only weakly polytime sound, where, in particular, we allow only a *lazy* cut-elimination procedure not reducing the so-called additive commutative cuts [8].

Asperti's Intuitionistic Light Affine Logic [1] works well as a calculus of intuitionistic proof-nets or as Terui's term calculus[16] (the latter enjoys polytime strong normalization), but due to unrestricted weakening cut elimination becomes non-deterministic (see Example 8.1).

By invoking the full capacity of the second-order systems, Mairson and Terui [12] show that polytime computation can be expressed within the second-order Multiplicative fragment of linear logic, but their encoding is extremely complicated.

We summarize these pros and cons in Table 1.

[1] Here we invoke the 'additive' connective $\oplus$ with its inference rules (5).

Table 1
A 'missing link'.

System	Expressive power vs. Cut-elimination complexity
Girard's LLL	"We shall define a *lazy* cut-elimination which terminates in polytime. The result of the procedure is cut-free only in certain cases, but this is enough for us." Girard [8].
Asperti's ILAL	Additive-free Intuitionistic Light Affine Logic works well as a system to represent polynomial-time computations. But cut elimination becomes non-deterministic (see Example 8.1).
"$\mathcal{L}$ **???**"	The system should easily simulate all programming tricks we need, and enjoy a clear polytime strong normalization and the Church–Rosser property.
MLL	The second-order Multiplicative fragment of linear logic is capable of expressing polytime computation (Mairson and Terui [12]), but their encoding is extremely complicated.

The aim of our research is to find a 'missing link' to fill the 'ideal' third line in Table 1 — that is to find a logical system $\mathcal{L}$ such that:

(a) $\mathcal{L}$ is powerful enough to represent each f from a given complexity class $\mathcal{C}$:
  (a1) Ideally, $\mathcal{L}$ should incorporate basic constructs (such as conditionals, iteration, etc.) to simulate computations in a natural way.

(b) For any f represented in $\mathcal{L}$, the corresponding cut-elimination procedure is of low complexity to provide that $f \in \mathcal{C}$:
  (b1) Ideally, $\mathcal{L}$ should enjoy a transparent fast strong normalization (to guarantee an upper complexity bound) and the Church–Rosser property (to provide deterministic results).

1.2. Summary

Light linear logic has been designed as a specific formal system to accommodate the following principles for modalities ! and § [8]:

$$!A \vdash (!A \otimes !A);\ !A \vdash 1;\ !(A \& B) = (!A \otimes !B);\ \frac{A \vdash B}{!A \vdash !B};\ !A \vdash \S A;\ \frac{\Gamma \vdash C}{\S\Gamma \vdash \S C} \tag{3}$$

As compared to traditional sequent calculi, the syntax of LLL is much more complicated. At the same level of formulas, it invokes also 'blocks' of formulas, such as a 'discharged formula' $[A]$ and a 'comma expression' $\langle A_1, A_2, \ldots, A_\ell \rangle$.

As a result, the application of the traditional cut-elimination argument for LLL sequent calculus runs into essential technical difficulties, even within the framework of proofnets.

The aim of the paper is to simplify LLL, resulting in a traditional sequent calculus, called Easy-LLL, that meets the fundamental complexity-theoretic constraints of the original LLL, and, on the other hand, enjoys polynomial-time strong normalization and the Church–Rosser property, and the strong representability of polytime computation.

The original syntax of LLL is too complicated, mainly because one has to deal with sequents which do not just consist of formulas but also of 'blocks' of formulas.

We circumvent the complications of 'blocks' by introducing a new modality ∇ which is exclusively in charge of 'additive blocks'. The most interesting feature of this purely multiplicative ∇ is the possibility of the second-order encodings of additive connectives.

The resulting system (with the traditional syntax), called Easy-LLL, is still powerful to represent any deterministic polynomial time computations in purely logical terms.

Unlike the original LLL and the classical version of light affine logic, Easy-LLL admits polynomial time strong normalization, namely, cut elimination terminates in a *unique way* in polytime by any choice of cut reduction strategies.

2. Light logics: a naïve timed resource semantics

As an underlying logical formalism, we use *linear logic* introduced by Girard [5] as a *resource-sensitive* refinement of the traditional logic.

In turn, we will interpret *light linear logic* as a *timed resource-sensitive* refinement of linear logic.

Syntactic peculiarities of light logics are considered in Section 3.

2.1. 'Multiplicative' and 'Additive' connectives: basic principles

Within linear logic,

(i) the traditional conjunction $\wedge$ is split into two connectives: $\otimes$ (*tensor*) and & (*with*), and,
(ii) dually, the traditional disjunction $\vee$ is split into two connectives: $\parr$ (*par*) and $\oplus$ (*plus*).

Table 2
The Second-order Multiplicative-Additive fragment of linear logic [5]. We use here a one-side calculus, where *linear negation* $A^{\perp}$ is used as an abbreviation in the sense of the de Morgan dual, except for atomic formulas $p^{\perp}$.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp},\ A}$(identity)	$\dfrac{\vdash \Gamma,\ A \qquad \vdash A^{\perp},\ \Theta}{\vdash \Gamma,\ \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma,\ A \quad \vdash \Theta,\ B}{\vdash \Gamma,\ \Theta,\ (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma,\ A,\ B}{\vdash \Gamma,\ (A \parr B)}$(par) $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma,\ \bot}$(bottom)
Additives:	
$\dfrac{\vdash \Gamma,\ A \quad \vdash \Gamma,\ B}{\vdash \Gamma,\ (A\&B)}$(with) $\quad \dfrac{}{\vdash \Gamma,\ \top}$(true)	$\dfrac{\vdash \Gamma,\ A}{\vdash \Gamma,\ (A \oplus B)}$(plus$_1$) $\dfrac{\vdash \Gamma,\ B}{\vdash \Gamma,\ (A \oplus B)}$(plus$_2$)
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma,\ A(p)}{\vdash \Gamma,\ \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma,\ A(B)}{\vdash \Gamma,\ \exists p A(p)}$(there-exists)

From the resource point of view, $(A \otimes B)$ means that:

"*Both 'resources' A and B are available concurrently but in a non-shared manner*",

while $(A\&B)$ stands for:

"*Any 'resource' of A and B is available*."

Accordingly, the so-called 'multiplicative' $\otimes$ is formally defined by the following inference 'non-shared' rules (see Table 2):

$$\mathbf{L}\otimes\ \frac{\Gamma, A, B \vdash \Sigma}{\Gamma, (A \otimes B) \vdash \Sigma} \qquad \mathbf{R}\otimes\ \frac{\Gamma_1 \vdash A, \Sigma_1 \qquad \Gamma_2 \vdash B, \Sigma_2}{\Gamma_1, \Gamma_2 \vdash (A \otimes B), \Sigma_1, \Sigma_2}$$

while the 'additive' connective & is formally defined by the rules with 'shared resources':

$$\frac{\Phi,\ B_1 \vdash \Sigma}{\Phi,\ (B_1\&B_2) \vdash \Sigma}\ \mathbf{with}_1 \qquad \frac{\Phi,\ B_2 \vdash \Sigma}{\Phi,\ (B_1\&B_2) \vdash \Sigma}\ \mathbf{with}_2 \qquad \frac{\Gamma \vdash B_1,\ \Sigma \qquad \Gamma \vdash B_2,\ \Sigma}{\Gamma \vdash (B_1\&B_2),\ \Sigma}\ \mathbf{R\text{-}with} \tag{4}$$

The rules for $\parr$ and $\oplus$ are introduced in a straightforward dual manner.

In particular, for $\oplus$ we have the following inference rules:

$$\frac{\Gamma,\ B_1 \vdash \Sigma \qquad \Gamma,\ B_2 \vdash \Sigma}{\Gamma,\ (B_1 \oplus B_2) \vdash \Sigma}\ \mathbf{L\text{-}plus} \qquad \frac{\Phi \vdash B_1,\ \Sigma}{\Phi \vdash (B_1 \oplus B_2),\ \Sigma}\ \mathbf{plus}_1 \qquad \frac{\Phi \vdash B_2,\ \Sigma}{\Phi \vdash (B_1 \oplus B_2),\ \Sigma}\ \mathbf{plus}_2 \tag{5}$$

2.2. 'Exponential' modalities: basic principles of ! and its dual ?

Within the traditional resource interpretation of linear logic [5–7], $!A$ is intended to express that:

"*Any number of copies of 'resource' A is available (at the current moment)*".

Here we modify such a resource interpretation by adding a *'time' direction*.
Namely, $!A$ is supposed to mean that:

"*Any number of copies of 'resource' A is available but at the next moment*".

Based on our timed resource interpretation, we can easily justify the following basic principles of ! introduced by Girard [8]:

- The *functorial* property:

$$\frac{A \vdash B}{!A \vdash !B} \tag{6}$$

- The *'exponential isomorphism'*:

$$!A \otimes !B = !(A\&B) \tag{7}$$

As a corollary, we get the following *Contraction* rule:

$$!A \otimes !A = !(A\&A) = !A \tag{8}$$

- The fact that

  "*Any number of the zero resources is available at any moment*"

  is expressed as:

$$\vdash !1 \tag{9}$$

  As a corollary, we get the following *weak version* of the 'exponential isomorphism':

$$!A \vdash !A \otimes !1 = !(A \& 1) \vdash !A, \quad \text{and, hence,} \quad !(A \& 1) = !A \tag{10}$$

- In turn, a pre-condition of the form:

  "*Any number of the zero resources is available at the next moment*,"

  can be conceived of as the 'garbage', which may be ignored by acceptance of the following principle:

$$!1 \vdash 1 \tag{11}$$

  As a corollary, here we get the following *Weakening* rule:

$$!A = !(A \& 1) \vdash !1 \vdash 1 \tag{12}$$

Remark 2.1. As for the corresponding *multi-functorial* property expressed as:

$$(!A \otimes !B) \vdash !(A \otimes B) \tag{13}$$

we follow Girard [8] and consider two versions:
(a) *Light linear logic* (LLL), in which principle (13) is not accepted.
(b) *Elementary linear logic* (ELL), in which principle (13) is accepted.

2.3. 'Paragraph' modalities: basic principles of § and its dual $§^{\perp}$

In order to formally represent the unary case of the exponentials, an additional modality § has been introduced in Girard [8][2]

Our proposal is to interpret a formula of the form §A as:

"*One copy of 'resource' A is available but at the next moment*".

Such a timed resource approach allows us to sort out the mystery of § and justify the following basic principles of § introduced by Girard [8]:

- The *functorial* property:

$$\frac{A \vdash B}{§A \vdash §B} \tag{14}$$

- The fact that

  "*The zero resource is available at any moment*"

  is expressed as:

$$\vdash §1 \tag{15}$$

- The *multi-functorial* property:

$$(§A \otimes §B) \vdash §(A \otimes B) \tag{16}$$

- The fact that

  $!A$ provides *at least one copy of A at the next moment*

  is expressed as the following *dereliction* principle:

$$!A \vdash §A \tag{17}$$

2.4. 'Nabla' modalities: basic principles of ∇ and its dual △

According to principle (12), the exponentials control weakening but *at the next moment*. In order to control weakening *at the current moment*, we introduce an additional modality ∇.

[2] "This strange connective has been introduced to compensate two things, namely the want of dereliction, but also the failure of the principle [V]: $(!A \otimes !B) \vdash !(A \otimes B)$, which is essential in the representation of data types." (Girard [8]).

Table 3
LLL-∇: The nabla-version of LLL. ∇ and $\triangle$ are duals of each other. $\S$ and $\S^{\perp}$ are duals of each other. A block of the form $\langle D\rangle$ is identified with the formula $\triangle D$.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp},\ A}$(identity)	$\dfrac{\vdash \Gamma,\ A \quad \vdash A^{\perp},\ \Theta}{\vdash \Gamma,\ \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma,\ A \quad \vdash \Theta,\ B}{\vdash \Gamma,\ \Theta,\ (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma,\ A,\ B}{\vdash \Gamma,\ (A \parr B)}$(par) $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma,\ \bot}$(bottom)
Additives:	
$\dfrac{\vdash \Gamma,\ A \quad \vdash \Gamma,\ B}{\vdash \Gamma,\ (A \& B)}$(with) $\quad \dfrac{}{\vdash \Gamma,\ \top}$(true)	$\dfrac{\vdash \Gamma,\ A}{\vdash \Gamma,\ (A \oplus B)}$(plus$_1$) $\dfrac{\vdash \Gamma,\ B}{\vdash \Gamma,\ (A \oplus B)}$(plus$_2$)
Structure/Modalities:	
$\dfrac{\vdash \Gamma,\ A_i}{\vdash \Gamma,\ \langle A_1, \ldots, A_i, \ldots, A_\ell\rangle}$(**A**-dereliction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma,\ \langle A_1, \ldots, A_\ell\rangle}$(**A**-weakening, $\ell \geq 1$)
$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ C}{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ \nabla C}$($\nabla$-rule, $n \geq 0$)	$\dfrac{\vdash \langle D_1, \ldots, D_\ell\rangle,\ C}{\vdash ?D_1,\ ..,\ ?D_\ell,\ !C}$ (!-rule, $\ell \geq 0$)
$\dfrac{\vdash \Gamma,\ ?A,\ ?A}{\vdash \Gamma,\ ?A}$(?-Contraction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma,\ ?A}$(?-Weakening)
$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ A_1,\ ..,\ A_m,\ C}{\vdash ?D_1,\ ..,\ ?D_n,\ \S^{\perp}A_1,\ ..,\ \S^{\perp}A_m,\ \S C}$ ($\S$-rule, $n, m \geq 0$)	
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma,\ A(p)}{\vdash \Gamma,\ \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma,\ A(B)}{\vdash \Gamma,\ \exists p A(p)}$(there-exists)

We will interpret a formula of the form ∇A as:

"*One copy of 'resource' A is available at the current moment, but it may be ignored*".

Based on such a timed resource interpretation, we state the following basic principles of ∇:

- The *functorial* property:

$$\frac{A \vdash B}{\nabla A \vdash \nabla B} \tag{18}$$

- The fact that "*The zero resource is available at any moment, and it may be ignored*" is expressed as:

$$\vdash \nabla 1 \tag{19}$$

- The *multi-functorial* property:

$$(\nabla A \otimes \nabla B) \vdash \nabla(A \otimes B) \tag{20}$$

- *Dereliction* and *Weakening* rules of the form:

$$\nabla A \vdash A, \qquad \nabla A \vdash 1 \tag{21}$$

- The *idempotent/projection/promotion* property:

$$\nabla\nabla A = \nabla A \tag{22}$$

3. Syntax for light logics: problems and solutions

In this section we design two syntactical systems:

- LLL-∇ within Table 3, to accommodate all the principles for *light linear logic* (see Section 2).
- ELL-∇ within Table 4, to accommodate all the principles for *elementary linear logic* (see Section 2).

To contract a number of inference rules, we will prefer one-side calculi, within which *linear negation* $A^{\perp}$ is used as an abbreviation in the sense of the de Morgan dual, except for atomic formulas $p^{\perp}$. We also omit the *permutation rules*, since we will deal with multisets of formulas and 'blocks' of formulas.

The system in Table 4 is given in the form of a traditional sequent calculus.

Table 4
ELL-∇: The nabla-version of ELL. ∇ and △ are duals of each other. § and $\S^{\perp}$ are duals of each other.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp},\ A}$(identity)	$\dfrac{\vdash \Gamma,\ A \quad \vdash A^{\perp},\ \Theta}{\vdash \Gamma,\ \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma,\ A \quad \vdash \Theta,\ B}{\vdash \Gamma,\ \Theta,\ (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma,\ A,\ B}{\vdash \Gamma,\ (A \parr B)}$(par) $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma,\ \perp}$(bottom)
Modalities:	
$\dfrac{\vdash \Gamma,\ A}{\vdash \Gamma,\ \triangle A}$(△-dereliction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma,\ \triangle A}$(△-weakening)
$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ C}{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ \nabla C}$(∇-rule, $n \geq 0$)	$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_\ell,\ C}{\vdash ?D_1,\ ..,\ ?D_\ell,\ !C}$ (!-rule, $\underline{\ell \geq 0}$)
$\dfrac{\vdash \Gamma,\ ?A,\ ?A}{\vdash \Gamma,\ ?A}$(?-Contraction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma,\ ?A}$(?-Weakening)
$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ A_1,\ ..,\ A_m,\ C}{\vdash ?D_1,\ ..,\ ?D_n,\ \S^{\perp}A_1,\ ..,\ \S^{\perp}A_m,\ \S C}$ (§-rule, $n, m \geq 0$)	
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma,\ A(p)}{\vdash \Gamma,\ \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma,\ A(B)}{\vdash \Gamma,\ \exists p A(p)}$(there-exists)

The syntax of the system in Table 3 is more complicated.
At the same level of formulas, we invoke also 'blocks' of formulas, such as an '**A**-expression'

$$\langle A_1, A_2, \ldots, A_\ell \rangle.$$

In Table 3, the idea behind

$$\langle A_1, A_2, \ldots, A_\ell \rangle$$

is to simulate

$$\triangle(A_1 \oplus A_2 \oplus \cdots \oplus A_\ell).$$

Accordingly, we will identify a *block* of the single form

$$\langle A \rangle$$

with the *formula*

$$\triangle A.$$

Remark 3.1. Within our systems, the basic principle:

$$\frac{A \vdash B}{!A \vdash !B},$$

is provided as a compound rule of the form:

$$\frac{\dfrac{A \vdash B}{\nabla A \vdash B}}{!A \vdash !B}$$

Remark 3.2. We do not include the additives rules in our Table 4, since the additives: & and ⊕, can be introduced as the following abbreviations, with mimicking the additive rules of Table 2:

$(A \,\&\, B) = \exists p\,(p \otimes \nabla(p \multimap A) \otimes \nabla(p \multimap B))$; $\quad \top = \exists p\, p$;
$(A \oplus B) = \forall p\,((\nabla(A \multimap p) \otimes \nabla(B \multimap p)) \multimap p)$; $\quad \mathbf{0} = \forall p\, p$.

As for Table 3, we need the explicit additive rules to deal with sequents that includes non-formula parts such as $\vdash \langle A^{\perp}, B^{\perp} \rangle$, $(A \& B)$ in Example 3.1. □

3.1. 'Nabla' modalities to save cut-elimination

One could say that the paragraph modality § is already somewhat mysterious, so that the presence of an extra pair of nabla modalities: ∇ and $\triangle$, makes things more complicated.

In this section we show that something like our ∇ is a necessary ingredient of the syntax to provide cut-elimination effects.

Example 3.1. When we are looking for appropriate sequent calculi for LLL and ELL, the most delicate issue is to provide the following direction of the *'exponential isomorphism'* (7):

$$!A,\ !B \vdash !(A\&B) \tag{23}$$

and, at the very least, its *weak version*:

$$!A \vdash !(A\&1) \tag{24}$$

Within our LLL-∇, we have a clear solution to (23):

$$\dfrac{\dfrac{\vdash A^{\perp},\ A}{\vdash \langle A^{\perp}, B^{\perp}\rangle,\ A} \qquad \dfrac{\vdash B^{\perp},\ B}{\vdash \langle A^{\perp}, B^{\perp}\rangle,\ B}}{\dfrac{\vdash \langle A^{\perp}, B^{\perp}\rangle,\ (A\&B)}{\vdash ?A^{\perp},\ ?B^{\perp},\ !(A\&B)}}$$

and a clear solution to (24):

$$\dfrac{\dfrac{\vdash A^{\perp},\ A}{\vdash \triangle A^{\perp},\ A} \qquad \dfrac{\vdash 1}{\vdash \triangle A^{\perp},\ 1}}{\dfrac{\vdash \triangle A^{\perp},\ (A\&1)}{\vdash ?A^{\perp},\ !(A\&1)}}$$

Proposition 3.1. *Even in the case of elementary linear logic ELL with additive rules, we need an additional modality like 'nabla' to provide a cut-free proof for a weak version of* (23)*:*

$$!p \vdash !(p\&1) \tag{25}$$

Proof. Along the lines of Girard [8], let ELL_0 be a nabla-free version of Table 4 enriched with additive rules from Table 2, and with an !-rule of the form

$$\dfrac{\vdash D_1,\ ..,\ D_{\ell},\ C}{\vdash ?D_1,\ ..,\ ?D_{\ell},\ !C}$$

With such a system, any candidate for a cut-free proof for a sequent of the form

$$\vdash ?p^{\perp},\ !(p\&1),$$

fails, since the proof must be like this:

$$\dfrac{\vdash p^{\perp},\ p \qquad \dfrac{???}{\vdash p^{\perp},\ 1}}{\dfrac{\vdash p^{\perp},\ (p\&1)}{\vdash ?p^{\perp},\ !(p\&1)}}$$

Hence, to save cut-elimination for light linear logic, as well as for elementary linear logic, we need, at the least, '**A**-weakening' of the form

$$\dfrac{\vdash 1}{\vdash \triangle p^{\perp},\ 1}$$ □

4. Phase semantics for LLL-∇ and ELL-∇

Along the lines of [9], we introduce a fibred (stratified) phase semantics for LLL-∇.

The basic idea is to interpret any formula of the new form ∇A as:

$$(\nabla A)^{*} = (A^{*} \cap J)^{\perp\perp},$$

where J is a fixed submonoid of a monoid M such that $J \subseteq \{e\}^{\perp\perp}$; here e denotes the neutral element of M.

A *fibred phase space* is introduced as a sequence of phase spaces $\mathcal{M}_0, \mathcal{M}_1, \ldots$, providing a certain one-way interaction between adjacent $\mathcal{M}_{n+1}$ and $\mathcal{M}_n$ by means of functions h_n and f_n. These functions provide the evaluation of $!A$ and §A by referencing to *the next moment/level*.

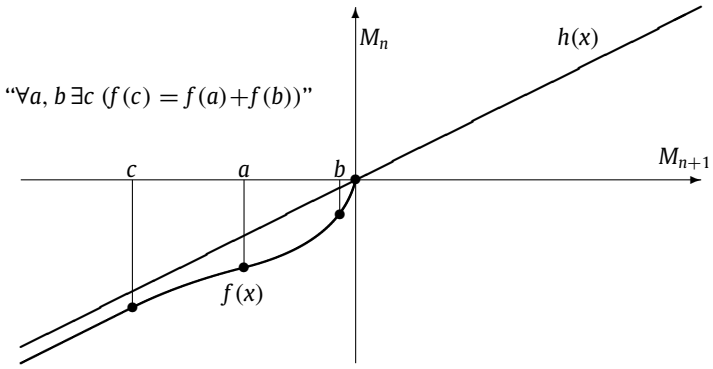


Fig. 1. The intermediate value property. Here M_n and M_{n+1} are monoids of reals, with $+$ as a monoidal operation, 0 is the neutral element, $x \preceq y$ is defined as: $x \leq y$. f is an arbitrary continuous function bounded by a linear function h.

Definition 4.1. A *phase space* $(M, e, \bot)$ is a commutative monoid M with a distinguished subset $\bot \subseteq M$; here e stands for the neutral element of M.

For any subset $X \subseteq M$, define:

$$X^{\bot} = \{y \in M \mid \forall x \in X\ (xy \in \bot)\}.$$

A subset $X \subseteq M$ is *closed* iff $X^{\bot\bot} = X$. In particular, let $\mathbf{1}$ be the subset $\{e\}^{\bot\bot}$.

The phase space induces a natural preorder on the underlying monoid compatible with monoid multiplication:

$$x \preceq y \Leftrightarrow x \in \{y\}^{\bot\bot}.$$

Definition 4.2. A *homomorophism of phase spaces*, or simply a *phase homomorphism*, is a monoid homomorphism

$$h : M \to M'$$

such that $h(\bot) \subseteq \bot'$.

Given a partial mapping

$$f : M \to M',$$

we say that f is *bounded by* h, if $f(a) \preceq h(a)$ for every $a \in J$, where J is the domain of f.

The mapping f has *the intermediate value property* if for every a and b from its domain J, there exists $c \in J$ such that $c \preceq a$ and $c \preceq b$, and $f(a)f(b) = f(c)$, (see Fig. 1). □

Proposition 4.1. *The class of functions with the intermediate value property is quite natural:*

- *Any monoid homomorphism $f : \mathbf{1} \to M$ has the intermediate value property.*
- *In Fig. 1 we show an example of a function f such that f has the intermediate value property, but f is not a monoid homomorphism from $\mathbf{1}_{n+1}$ into M_n.*

Definition 4.3. A *fibred phase space* is a sequence

$$((M_n, e_n, \bot_n),\ h_n, J_n, f_n, L_n)_{n \geq 0},$$

where for each n,

(a) $(M_n, e_n, \bot_n)$ is a phase space,
(b) $h_n : M_{n+1} \to M_n$ is a phase homomorphism,
(c) J_n is a submonoid of $\mathbf{1}_n$,
(d) $f_n : J_{n+1} \to M_n$ is a mapping with the intermediate value property such that f_n is bounded by h_n, and, in addition, $f_n(e_{n+1}) = e_n$ (see Fig. 1).
(e) L_n is a submonoid of $\mathbf{1}_n$ such that every element a of L_n is a *weak idempotent,* i.e., $a \preceq_n a \cdot_n a$.

Definition 4.4. A *fibred phase space*

$$((M_n, e_n, \bot_n),\ h_n, J_n, f_n, L_n)_{n \geq 0}$$

is said to be an *ELL-fibred phase space* if each of the functions f_n happens to be a monoid homomorphism from J_{n+1} to M_n (cf. Proposition 4.1).

Definition 4.5. Let

$$((M_n, e_n, \perp_n), h_n, J_n, f_n, L_n)_{n\geq 0}$$

be a fibred phase space.

Given an assignment of closed sets $p^{*0}, p^{*1}, \ldots, p^{*n}, \ldots$ to propositional atoms p, for each formula A we associate a sequence of closed sets $A^{*0}, A^{*1}, \ldots, A^{*n}, \ldots$, in the following inductive way:

$$1^{*n} = \mathbf{1}_n = \{e_n\}^{\perp_n\perp_n}, \qquad \top^{*n} = M_n,$$

$$(A \otimes B)^{*n} = (A^{*n} \cdot_n B^{*n})^{\perp_n\perp_n}, \qquad (A^{\perp})^{*n} = (A^{*n})^{\perp_n},$$

$$(A\&B)^{*n} = (A^{*n} \cap B^{*n}), \qquad (\nabla A)^{*n} = (A^{*n} \cap J_n)^{\perp_n\perp_n},$$

$$(\S A)^{*n} = (h_n(A^{*n+1}))^{\perp_n\perp_n}, \qquad (!A)^{*n} = (f_n(A^{*n+1} \cap J_{n+1}) \cap L_n)^{\perp_n\perp_n}.$$

For a 'block' of the form $\langle A_1, A_2, \ldots, A_\ell\rangle$, define:

$$\langle A_1, A_2, \ldots, A_\ell\rangle^{*n} = ((A_1^{*n} \cup A_2^{*n} \cup \cdots \cup A_\ell^{*n})^{\perp_n} \cap J_n)^{\perp_n}.$$

The second-order case is considered as in [9].

A formula A is *valid* in the fibred phase space, if for each n, $e_n \in A^{*n}$ in any valuation $p^{*0}, p^{*1}, \ldots$, to propositional atoms p. □

Theorem 4.1 (*Strong Completeness*)**.**

(a) *If a formula A is provable in LLL-∇, then A is valid in any fibred phase space.*
(b) *If a formula A is valid in any fibred phase space, then A is provable in LLL-∇ without the cut rule.*

Corollary 4.1 (*Cut-Elimination*)**.** *Theorem 4.1 provide a semantic proof for cut elimination in LLL-∇.*

Remark 4.1. Corollary 4.1 cannot give more than the pure existence of cut-free proofs.

As for the traditional cut-reduction arguments where each of the cuts is reduced to a number of cuts by dealing only with a pair of inference rules, the problematic case caused by the full version of '!-rule' is given in Example 4.1.[3]

Example 4.1. The cut below cannot be eliminated by the traditional argument mentioned above:

$$\dfrac{\dfrac{\dfrac{\cdots}{\vdash \langle A^{\perp}\rangle,\ B}}{\vdash ?A^{\perp},\ !B} \qquad \dfrac{\dfrac{\cdots}{\vdash \langle B^{\perp}, C^{\perp}\rangle,\ D}}{\vdash ?B^{\perp},\ ?C^{\perp},\ !D}}{\vdash ?A^{\perp},\ ?C^{\perp},\ !D}\ (\text{cut})$$

Remark 4.2. Nevertheless, with certain modifications to handle **A**-blocks, we can define a syntactical strongly normalizing cut-elimination procedure for LLL-∇.

Proof Sketch. We treat a block $\langle\Gamma\rangle$ as a 'generalized △-formula', and replace '∇-rule' in Table 2 with:

$$\frac{\vdash \langle\Gamma_1\rangle,\ \ldots,\ \langle\Gamma_n\rangle,\ C}{\vdash \langle\Gamma_1\rangle,\ \ldots,\ \langle\Gamma_n\rangle,\ \nabla C}\ (\nabla\text{-rule}', n\geq 0) \tag{26}$$

To capture strange cuts between ∇B and the 'generalized △-formula' that includes $B^{\perp}$, we add cut rules like this:

$$\frac{\vdash \langle\Gamma\rangle,\ \nabla B \qquad \vdash \langle B^{\perp}, \Phi\rangle,\ \Theta}{\vdash \langle\Gamma, \Phi\rangle,\ \Theta}\ (\text{cut}') \tag{27}$$

Accordingly, the cut in the problematic case of Example 4.1 will be reduced to:

$$\dfrac{\dfrac{\dfrac{\dfrac{\cdots}{\vdash \langle A^{\perp}\rangle,\ B}}{\vdash \langle A^{\perp}\rangle,\ \nabla B} \qquad \dfrac{\cdots}{\vdash \langle B^{\perp}, C^{\perp}\rangle,\ D}}{\vdash \langle A^{\perp}, C^{\perp}\rangle,\ D}}{\vdash ?A^{\perp},\ ?C^{\perp},\ !D}\ (\text{cut})$$

With additional cut reductions, we can provide termination but in *hyper-exponential* time, the height of the tower of exponents is determined by the depth of !-boxes. This huge bound is caused by the fact that we allow more than one conclusion of the form $?D_j$ within the "!-rule" (cf. Tables 3 and 5). □

4.1. Strong completeness for ELL-∇

Theorem 4.2 (*Strong Completeness*)**.**

(a) *If a formula A is provable in ELL-∇, then A is valid in any ELL-fibred phase space.*
(b) *If a formula A is valid in any ELL-fibred phase space, then A is provable in ELL-∇ without the cut rule.*

[3] Girard [8]: "We shall define a *lazy* cut-elimination which terminates in polytime. The result of the procedure is cut-free only in certain cases, but this is enough for us."

Table 5
The Inference Rules of Easy-LLL. ∇ and $\triangle$ are duals of each other. $\S$ and $\S^{\perp}$ are duals of each other.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp},\ A}$(identity)	$\dfrac{\vdash \Gamma,\ A \quad \vdash A^{\perp},\ \Theta}{\vdash \Gamma,\ \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma,\ A \quad \vdash \Theta,\ B}{\vdash \Gamma,\ \Theta,\ (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma,\ A,\ B}{\vdash \Gamma,\ (A \parr B)}$(par) $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma,\ \perp}$(bottom)
Modalities:	
$\dfrac{\vdash \Gamma,\ A}{\vdash \Gamma,\ \triangle A}$($\triangle$-dereliction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma,\ \triangle A}$($\triangle$-weakening)
$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ C}{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ \nabla C}$($\nabla$-rule, $n \geq 0$)	$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_{\ell},\ C}{\vdash ?D_1,\ ..,\ ?D_{\ell},\ !C}$ (!-rule, $\underline{\ell = 0,\ 1}$)
$\dfrac{\vdash \Gamma,\ ?A,\ ?A}{\vdash \Gamma,\ ?A}$(?-Contraction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma,\ ?A}$(?-Weakening)
$\dfrac{\vdash \triangle D_1,\ ..,\ \triangle D_n,\ A_1,\ ..,\ A_m,\ C}{\vdash ?D_1,\ ..,\ ?D_n,\ \S^{\perp}A_1,\ ..,\ \S^{\perp}A_m,\ \S C}$ ($\S$-rule, $n, m \geq 0$)	
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma,\ A(p)}{\vdash \Gamma,\ \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma,\ A(B)}{\vdash \Gamma,\ \exists p A(p)}$(there-exists)

Corollary 4.2 (*Cut-Elimination*)**.** *Theorem* 4.2 *provide a semantic proof for cut elimination in ELL-*∇.

Remark 4.3. We can show that syntactically ELL-∇ admits confluent strong normalization in elementary time — that is, cut elimination terminates in a unique way (Church–Rosser) in *hyper-exponential* time by any choice of cut reduction strategies, the height of the tower of exponents is determined by the depth of !-boxes.

Notice that ELL-∇ does not invoke additive rules, so that such a monstrous gap between the ELL-cut-elimination in elementary time and the lazy LLL-cut-elimination in polynomial time is caused *only* by the fact that we allow more than one conclusion of the form $?D_j$ within the "!-rule" (cf. Tables 4 and 5). □

5. The Easy fragment of LLL (Easy-LLL)

Here we introduce a traditional syntactical fragment of LLL-∇, called Easy-LLL, which is of our main interest.

The syntax of Table 3 is almost perfect: only one rule, the '!-rule', invokes a block of formulas of the form $\langle D_1, \ldots, D_{\ell}\rangle$ in its premise.

To get rid of blocks of formulas, we confine '!-rule' to $\ell \leq 1$. Recall that we identify a *block* of the single form $\langle D\rangle$ with the *formula* $\triangle D$.

In Table 5 we represent the resulting Easy fragment of LLL, which deals with formulas only.

Remark 5.1. By the same token of Remark 3.2, we exclude the additives rules in our Table 5, since the additive rules of Table 2 can be mimicked with the help of the following abbreviations:

$$(A \,\&\, B) = \exists p\,(p \otimes \nabla(p \multimap A) \otimes \nabla(p \multimap B)); \qquad \top = \exists p\, p;$$
$$(A \oplus B) = \forall p\,((\nabla(A \multimap p) \otimes \nabla(B \multimap p)) \multimap p); \qquad \mathbf{0} = \forall p\, p. \quad \square$$

Proposition 5.1. *Easy-LLL accommodates all principles for* light linear logic *given in Section* 2*:*

$$\frac{A \vdash B}{!A \vdash !B};\ (!A \otimes !A) = !A;\ !A \vdash 1;\ !A \vdash \S A;\ \frac{\Gamma \vdash C}{\S\Gamma \vdash \S C};\ !(A \& B) \vdash (!A \otimes !B);\ !A = !(A \& 1);$$

except for the principle: $(!A \otimes !B) \vdash !(A \& B)$.

Remark 5.2. From the provability point of view, Easy-LLL is weaker than LLL-∇.

One of the basic principles in LLL [8] is the *'exponential isomorphism'* (7):

$$(!p \otimes !q) = !(p \& q)$$

One direction: $!(p \& q) \vdash (!p \otimes !q)$, is easily provable in Easy-LLL.

As for other direction: $(!p \otimes !q) \vdash !(p \& q)$, it is provable in LLL-$\nabla$ (for the cost of its 'mixed' syntax with **A**-blocks), but it is not provable in Easy-LLL.

It is unclear how to incorporate

$$(!p \otimes !q) \vdash !(p \& q)$$

into a traditional cut-free sequent calculus formalism within the constraints of light linear logic. For instance, by allowing $\ell = 2$ in the "!-rule" in Table 5, we 'jump' to ELL, which encodes *elementary functions* computable in *hyper-exponential time*.

On the other hand, what we actually need to represent polytime computation in [8,2] is a weaker form of the 'exponential isomorphism':

$$(!p \otimes !1) = !(p \& 1),$$

which is provable in Easy-LLL, as well. □

Corollary 5.1. *We can show that Easy-LLL is sound and complete with respect to the* fibred phase semantics *obtained from Definition 4.3 by omitting the requirement that "f_n has the intermediate value property".*

In Section 6 we show that Easy-LLL enjoys *strong normalization in polytime*, providing a *unique cut-free form*. Namely, any proof is normalizable in s^d steps with one and the same result regardless of which sequence of cut reductions we take (here s is the size of the proof, and the degree d is determined only by the nesting depth of exponentials in the proof).

In Section 7 we show that Easy-LLL has the full expressive power of light logics, and captures exactly polytime computation.

6. Easy-LLL: confluent strong normalization in polytime

In this section we will prove polytime strong normalization for Easy-LLL.

Definition 6.1. In full accordance with the rules from Table 5, any proof within Easy-LLL can be represented as a *proofnet* π (cf. [8]), which consists of nodes and boxes, some of them are connected by arrows, so that

(a) An axiom of the form: $\vdash A^{\perp}, A$, is represented with a node labeled by "axiom" such that it has only two outgoing arrows labeled by formulas of the form A and $A^{\perp}$, resp.

The *cut* rule is represented with a node labeled by "cut" such that it has only two incoming arrows labeled by formulas of the form A and $A^{\perp}$, resp.

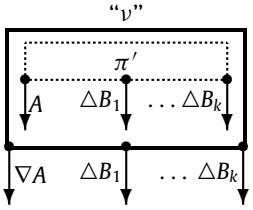
(b) The *times* and *par* rules are represented with nodes labeled by "$\otimes$" or "$\parr$" such that the nodes have exactly two incoming arrows, labeled by some A and B, and one outgoing arrow labeled by $(A \otimes B)$ or $(A \parr B)$, resp.

The *one* and *bottom* rules are represented with nodes labeled by "1" or "$\perp$" such that the nodes have only an outgoing arrow labeled by 1 or $\perp$.

(c) The ($\triangle$-dereliction) rule is represented with a node labeled by "$\triangle$" such that it has one incoming arrow, labeled by an A, and one outgoing arrow labeled by $\triangle A$.

The ($\triangle$-weakening) rule is represented with a node labeled by "w" such that it has one outgoing arrow labeled by a formula of the form $\triangle A$.

The (∇-rule) is represented with a node labeled by a ∇-box of the form (here π' is a proofnet with $k+1$ conclusions $A, \triangle B_1, \ldots, \triangle B_k$):

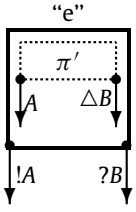


such that the node has $k+1$ outgoing arrows labeled by the corresponding conclusions: $\nabla A, \triangle B_1, \ldots, \triangle B_k$, of the ∇-box.

(d) The (?-contraction) rule is represented with a node labeled by "**C**" such that it has two incoming arrows, labeled by one and the same $?A$, and one outgoing arrow labeled by the same $?A$.

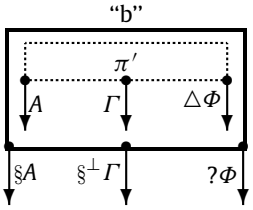
The (?-weakening) rule is represented with a node labeled by "W" such that it has one outgoing arrow labeled by a formula of the form $?A$.

The (!-rule) is represented with a node labeled by an !-box of the form (here π' is a proofnet with conclusions $A, \triangle B$):



such that the node has two outgoing arrows labeled by $!A$ and $?B$.

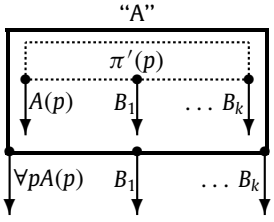
The (§-rule) is represented with a node labeled by a §-box of the form (here π' is a proofnet with conclusions A, Γ, $\triangle\Phi$, where Γ and Φ are multisets of formulas):



such that the node has outgoing arrows labeled by the corresponding conclusions of the §-box.

(e) The *there-exists* rule is represented with a node labeled by "∃" such that it has one incoming arrow, labeled by some $A(B)$, and one outgoing arrow labeled by $\exists pA(p)$.

The *for-all* rule is represented with a node labeled by a ∀-box of the form



such that the node has $k+1$ outgoing arrows labeled by the conclusions of the ∀-box. Here $\pi'(p)$ is a proofnet with a marked variable p, and each of the conclusions $B_1, \ldots, B_k$ has no free occurrences of p. The propositional variable p is considered to be *bound* within the ∀-box.

(f) Any unlabeled node has only an incoming arrow labeled by an A, which is said to be a *conclusion* of π. □

Definition 6.2. The *size* of a proofnet π is defined as the total number of all nodes and boxes contained inside π.

In order to control the size of proofnets under cut reductions (see Lemma 6.1), for any box E, we introduce an *adjusted size* $w(E)$, call it the *weight* of box E, as follows:

$$w(E) = s(\pi') + 2,$$

where $s(\pi')$ is the size of the proofnet π' inside E.

For a given nesting depth l of !-boxes and §-boxes inside π (we call it 'level l'), we take a 'slice' $\pi^{(l)}$ on *level* l as the collection of all nodes and boxes (together with arrows) on the depth l. The number of these nodes and boxes is called the *size of* π' *on level* l (each box is counted as a one node).

Definition 6.3. Exploring pairs of the inference rules in Table 5 gives rise to a natural set of cut reduction rules: we collect these reductions in Figs. 5–9.

Lemma 6.1. *We will list the basic peculiarities of our cut reduction rules:*

(a) *Each of the reductions does not mix levels.*

(b) *All reductions contract the size of a proofnet, except for* (∇, ∇), $(!, !)$, $(!, \S)$, $(\S, \S)$, $(!, C)$, *and* $(side, \forall)$.

(c) (∇, ∇) *and* $(side, \forall)$ *preserve the size of a proofnet (and even the names of boxes), but put the cut at hand inside the corresponding expanded "box*$_1$*". The effect is that this cut will not directly contact with "box*$_1$*" in future.*

(d) $(!, !)$, $(!, \S)$ *and* $(\S, \S)$ *merge "box*$_1$*" and "box*$_2$*" so that the weight of the new box (named as "box*$_1$*") does not exceed the sum of the weights of old boxes.*

(e) *Lastly,* $(!, C)$*-reductions do not change the number of* **C***-nodes.*

Each of the $(!, C)$*-reductions moves the corresponding* !*-box* E' *upward: from the position below a* **C***-node to the positions above the* **C***-node, with* E' *being doubled (see Fig. 3).*

With the help of Definition 6.4 and Lemma 6.2, we show that **C***-nodes form a forest structure, which is stable under cut reductions.*

Definition 6.4. We say that a **C**-node v_2 is a **C**-*son* of a **C**-node v_1, if these **C**-nodes are connected by an alternating chain of !-boxes and cuts as shown in Fig. 2, (the empty chain is allowed).

Lemma 6.2. *Let* π *be a proofnet constructed for a proof within Easy-LLL.*

Then the transitive closure of the relation "being a **C***-son" from Definition 6.4 is acyclic, and* **C***-nodes form a forest with respect to this relation.*

Moreover, given a level l *of* π*, the set of* **C***-nodes on level* l *remains intact by any of cut reductions on levels* $\geq l$*. Namely, if* v_2 *was a* **C**-son *of* v_1 *beforehand, then* v_2 *will be a* **C**-son *of* v_2 *after the cut reduction has been performed.*

Indeed, weakening cut reductions, such as $(\nabla, \triangle_w)$*-cut reduction, can remove some trees from the forest.*

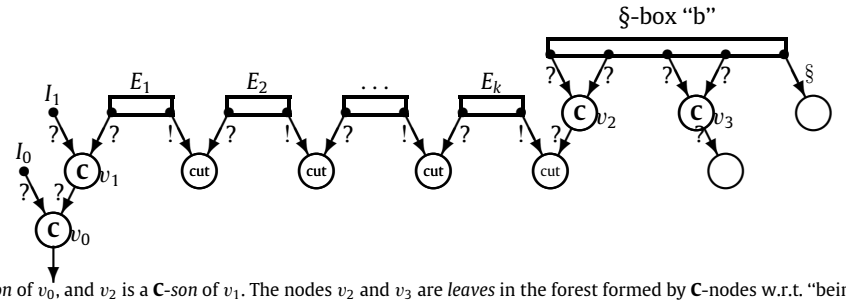


Fig. 2. Here v_1 is a **C**-*son* of v_0, and v_2 is a **C**-*son* of v_1. The nodes v_2 and v_3 are *leaves* in the forest formed by **C**-nodes w.r.t. "being a **C**-son".

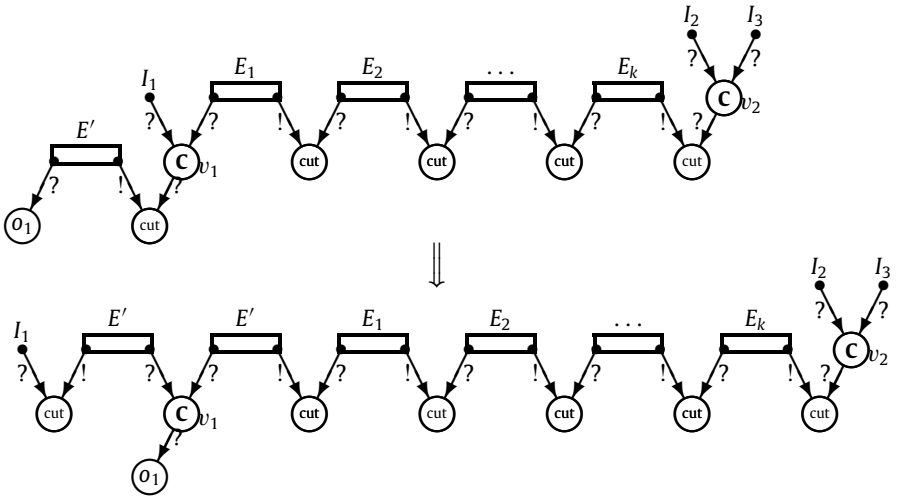


Fig. 3. An !-box E' moves through v_1, resulting in that a twin copy of E' appears on each of the branches of v_1.

Proof. The key case of $(!, C)$ reduction is shown in Fig. 3.

The (!, !), (!, §), and (§,§) reductions do not change the status of "being a **C**-son".

Other reductions cannot destroy "being a **C**-son".

Some reductions such as (!, §) reduction and $(\otimes, \parr)$ reduction might have had the effect of seemingly 'merging' different branches in the trees of **C**-nodes by means of a §-box "b" like in Fig. 2. But, according to Definition 6.4, which requires only !-boxes, the corresponding nodes v_2 and v_3 remain *leaves* in the forest of **C**-nodes. □

Theorem 6.1. *Let π be a proofnet constructed for a proof within Easy-LLL.*

Then whatever sequence of cut reductions we take, it terminates in s^d steps, with the size of any reduct being bounded by s^d, where s is the size of π, and the degree d is determined only by the maximum nesting depth of !-boxes inside π.

Proof. For each depth l, let:

- s_l be the size of the 'slice' $\pi^{(l)}$ on level l;
- n_l be the number of ∇-boxes and $\forall$-boxes on level l;
- e_l be the number of !-boxes on level l; and
- C_l be the number of **C**-nodes on level l.

Assume that cut reductions α and β are applied in a row (α, then β), and α is applied on level l, and β is applied on level l' such that $l' < l$. It is clear that these two reductions can commute, and the number of reductions does not decrease. For instance, for β being a (!,**C**)-cut reduction, the resulting sequence of reductions will be β, α, α.

Hence, the 'worst-case scenario' is, first, to reduce all cuts on level 0, then on level 1, then on level 2, and so on.

Let:

- T'_l denote the number of reduction steps on level l after having accomplished all reductions on level $l-1$.
- s'_l denote the size of the modified 'slice' on level l after having accomplished all reductions on level $l-1$.
- C'_l denote the number of **C**-nodes on level l after having accomplished all reductions on level $l-1$.

We start with a rough estimate for the number of cut reductions on level 0.

The case of size-contracting reductions is trivial.

Since a particular cut within (∇, ∇) and $(side, \forall)$ reductions cannot directly contact with the same "box$_1$" twice, the number of such reductions is bounded by $s_0 \cdot n_0$.

The most complicated case is related to $(!, C)$ reductions.

According to Lemma 6.1, we get a stable forest formed by **C**-nodes on level 0. In particular, the number of branches leading from a root to a non-branching **C**-node on level 0 is always bounded by C_0.

Each of the !-boxes that participate in $(!, C)$ reductions (see Fig. 3) will be doubled but only *one copy* of the !-box will be moved along a branch towards its non-branching **C**-node. Hence, at any reduction step on level 0 the number of !-boxes involved along one branch cannot exceed e_0.

Therefore, the size of $\pi^{(1)}$ on level 1 can be multiplied by the number of branches, at most. More precisely:

$$\begin{cases} s'_1 \le 2C_0 s_1 \\ C'_1 \le 2C_0 C_1 \end{cases} \tag{28}$$

Indeed, on the road towards the non-branching **C**-node, !-boxes can be merged by $(!, !)$-reductions, but this does not increase the total sum of their weights.

The maximum number of the $(!, !)$ and $(!, C)$ reductions that could have been involved in along one branch is bounded by $e_0^2 + e_0 C_0$.

Since the number of branches is bound by C_0, the total number of $(!, !)$ and $(!, C)$ reductions on level 0 is bounded by $2C_0(e_0^2 + e_0 C_0)$.

Bringing all contracting and non-contracting cases together, the number of cut reductions on level 0 is bounded as follows:

$$T'_0 \le s_0 + s_0 n_0 + 2C_0(e_0^2 + e_0 C_0) \le 2C_0 s_0^2 \tag{29}$$

By repeatedly applying the above procedure, we get the following bounds for cut reductions:

$$\begin{cases} s'_{l+1} \le 2C'_l s_{l+1} \\ C'_{l+1} \le 2^{l+1} C_0 C_1 C_2 \cdots C_{l+1} \\ T'_l \le 2C'_l (s'_l)^2 \end{cases} \tag{30}$$

The effect is that every sequence of cut reductions must terminate in

$$s^{O(d)} \text{ steps},$$

where s is the size of π, and d is the maximum nesting depth of !-boxes inside π. □

Theorem 6.2. *Our system of cut reduction rules is locally confluent.*

Proof. By examining all critical pairs of cut reductions. □

Corollary 6.1. *Our system of cut reductions has the Church–Rosser property. Therefore, every sequence of cut reductions terminates in polynomial time with a unique cut-free proof.*

Proof. This follows from Theorems 6.1 and 6.2. □

7. The expressive power of Easy-LLL

Corollary 7.1. *Any polytime computable function is representable as a proof within Intuitionistic Easy-LLL.*

Proof. This easily follows from Girard–Asperti–Roversi's result [8,15,2], as well as from Murawski–Ong's interpretation of Bellantoni–Cook's safe recursion [14]. □

Remark 7.1. Embedding of Intuitionistic Affine logics

In fact, Easy-LLL can easily accommodate all programming constructs developed for Intuitionistic Light Affine logic in [15,2].

Here we consider a version of Intuitionistic Light Affine logic in which weakening to the right is allowed to apply only to the axiom: $\bot \vdash$, so that a sequent of the form $\bot \vdash C$ can be considered as an additional axiom.

Any proof for $\Gamma \vdash \Phi$ in this restricted version of Intuitionistic Light Affine logic can be easily simulated by a proof for $\Gamma^{\&1} \vdash \Phi^{\&1}$ in Easy-LLL with the help of the following embedding:

$$\begin{aligned}
p^{\&1} &= (1\&p), & 1^{\&1} &= 1, \\
\bot^{\&1} &= \forall p\,(1\&p), & (A \otimes B)^{\&1} &= (1\&(A^{\&1} \otimes B^{\&1})), \\
(A \multimap B)^{\&1} &= (1\&(A^{\&1} \multimap B^{\&1})), & (A^{\bot})^{\&1} &= (1\&(A^{\&1} \multimap \bot^{\&1})), \\
(A\&B)^{\&1} &= (1\&(A^{\&1}\&B^{\&1})), & (A \oplus B)^{\&1} &= (1\&(A^{\&1} \oplus B^{\&1})), \\
(\S A)^{\&1} &= (1\&\S A^{\&1}), & (!A)^{\&1} &= (1\&!A^{\&1}), \\
(\forall p\, A(p))^{\&1} &= (1\&\forall p\, A(p)^{\&1}), & (\exists p\, A(p))^{\&1} &= (1\&\exists p\, A(p)^{\&1}).
\end{aligned}$$

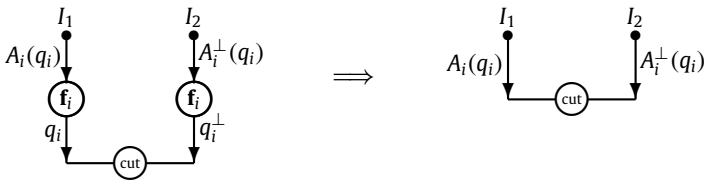


Fig. 4. The cut reduction in the case of fixed-point rules.

For instance, a specific choice for 'affinization' of $\perp$:

$$\perp^{\&1} = \forall p\,(1\&p),$$

provides that an affine proof of the form (C is an arbitrary formula):

$$\frac{\perp \vdash}{\perp \vdash C}$$

is simulated with a linear logic proof of the form:

$$\frac{(1\&C) \vdash (1\&C)}{\perp^{\&1} \vdash (1\&C)} \quad \square$$

Remark 7.2. In fact, Easy-LLL yields immediate simplifications.

For instance, we can take advantage of ∇ in Remarks 3.2 and 5.1 to provide "for free" the correct simulation of the additive boxes and related projections and conditionals within classical proofnets (cf. Girard [8]):

(1) Because of the Church–Rosser property, a proof of the form:

$$\dfrac{\dfrac{\dfrac{\pi_1}{\vdash \Gamma,\ B}}{\vdash \Gamma,\ (B\&B)} \qquad \dfrac{\dfrac{\pi_2}{\vdash \Gamma,\ B}}{\vdash \Gamma,\ (B\&B)} \qquad \dfrac{\dfrac{\pi_0}{\vdash \Phi,\ B^{\perp}}}{\vdash \Phi,\ (B^{\perp} \oplus B^{\perp})}\ \text{plus}_1}{\vdash \Gamma,\ \Phi}\ \text{cut}$$

is reduced to the same proof as its 'first projection':

$$\dfrac{\dfrac{\pi_1}{\vdash \Gamma,\ B} \qquad \dfrac{\pi_0}{\vdash \Phi,\ B^{\perp}}}{\vdash \Gamma,\ \Phi}\ \text{cut}$$

(2) A proof of the form:

$$\dfrac{\dfrac{\dfrac{\pi_1}{\vdash \Gamma,\ B}}{\vdash \Gamma,\ (B\&B)} \qquad \dfrac{\dfrac{\pi_2}{\vdash \Gamma,\ B}}{\vdash \Gamma,\ (B\&B)} \qquad \dfrac{\dfrac{\pi_0}{\vdash \Phi,\ B^{\perp}}}{\vdash \Phi,\ (B^{\perp} \oplus B^{\perp})}\ \text{plus}_2}{\vdash \Gamma,\ \Phi}\ \text{cut}$$

is reduced to the same proof as its 'second projection':

$$\dfrac{\dfrac{\pi_2}{\vdash \Gamma,\ B} \qquad \dfrac{\pi_0}{\vdash \Phi,\ B^{\perp}}}{\vdash \Gamma,\ \Phi}\ \text{cut}$$

Notice that ∇A itself can be conceived of as a 'projection', since

$$\nabla\nabla A = \nabla A$$

Remark 7.3. Theorem 6.1 gives the exact upper bound for the expressive power of Easy-LLL and its generalizations.

Corollary 7.2. *Let f be a function representable as a proof in Easy-LLL enriched with a number of fixed-point rules:*

$$\frac{\vdash \Gamma,\ A_i(q_i)}{\vdash \Gamma,\ q_i} \quad \textit{and} \quad \frac{\vdash \Gamma,\ A_i^{\perp}(q_i)}{\vdash \Gamma,\ q_i^{\perp}} \quad i = 1, 2, \ldots$$

where each of these $A_i(q_i)$ is a formula with a distinguished propositional variable q_i, the fixed-point of A_i.

Then f is computed in polynomial time. The degree of the polynomial is determined by the nesting depth of !*-boxes within the representation of f.*

Proof. Theorem 6.1 still remains valid, when we invoke the cut reductions for fixed-points in Fig. 4. $\square$

8. Concluding remarks

The 'nabla' modality has appeared here as a result of consecutive attempts to simplify the original 'mixed' syntax of Girard's *light linear logic* (LLL) and *elementary linear logic* (ELL) in accordance with the basic proof-theoretic and complexity constraints of LLL and ELL [8].

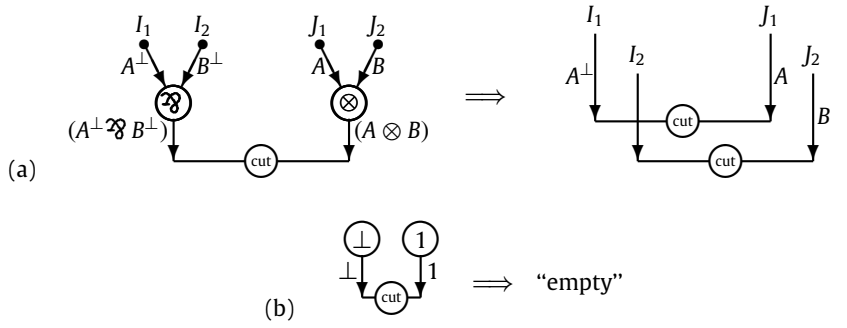


Fig. 5. The cut reduction rules for $(\otimes, \parr)$ and $(1, \bot)$.

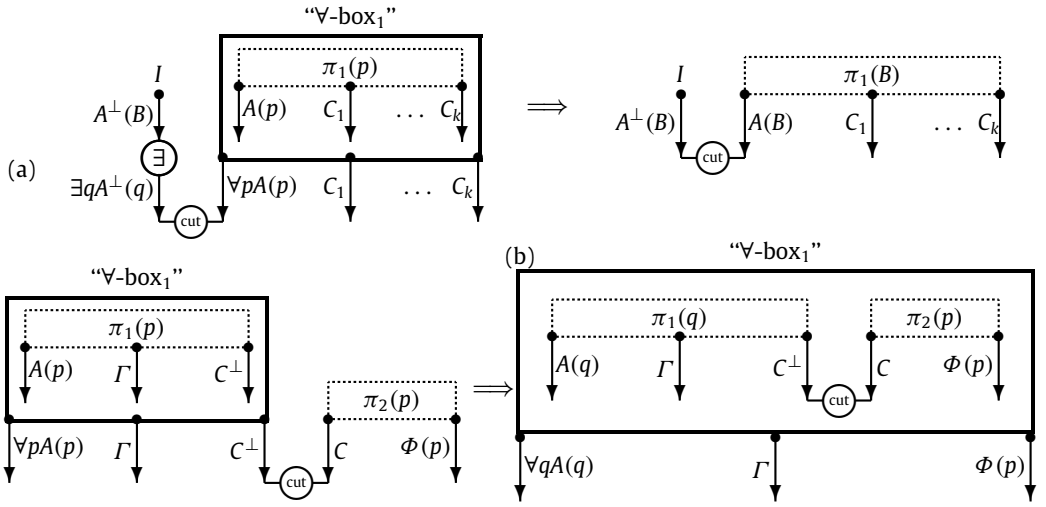


Fig. 6. (a) The $(\forall, \exists)$-cut reduction. (b) The (side, $\forall$)-cut reduction with renaming (q is fresh).

Remark 8.1. Based on our naïve resource semantics in Section 2, we have designed logical systems with three modalities where everything is under control:

(i) The § modality is a multi-functorial modality that raises a level:

$$A,\ B \vdash C \ \text{ yields } \ \S A,\ \S B \vdash \S C.$$

(ii) A functorial modality ! is applied to formulas on which Contraction and Weakening can apply but on the next level. The main observations about the morphisms this ! defines are

$$!A = (!A \otimes !A), \text{ and } !A \vdash 1, \text{ and } !A \vdash \S A.$$

See also Remark 5.2 about the *'exponential isomorphism'*.

(iii) The ∇ modality controls Weakening but on the current level.

In fact, our ∇A can be conceived of as a *purely multiplicative* connective that is served as the *greatest lower bound* of the *additive* $(A \& 1)$.

In particular, from the point of [6], ∇A can be interpreted as $!_1 A$.

One of the most interesting features of this purely multiplicative ∇ is the possibility of defining additive connectives without the need of unrestricted weakening (see Remarks 3.2 and 5.1):

$$(A \,\&\, B) = \exists p\, (p \otimes \nabla(p \multimap A) \otimes \nabla(p \multimap B)); \tag{31}$$

In fact, within linear logic itself [5], the additive connectives can be encoded using second order quantification, but these encodings crucially need the exponential modalities:

$$(A \,\&\, B) = \exists p\, (p \otimes\, !(p \multimap A) \otimes\, !(p \multimap B)) \tag{32}$$

Since ! enjoys weaker properties in LLL, the standard encoding (32), available in linear logic, does not work for light logics (not even for *elementary light logic*).

To express additives, light affine logic invokes unrestricted Weakening [2]. In presence of unrestricted weakening, as in affine logic, the additive connectives can be encoded as:

$$(A \,\&\, B) = \exists p\, (p \otimes (p \multimap A) \otimes (p \multimap B)) \tag{33}$$

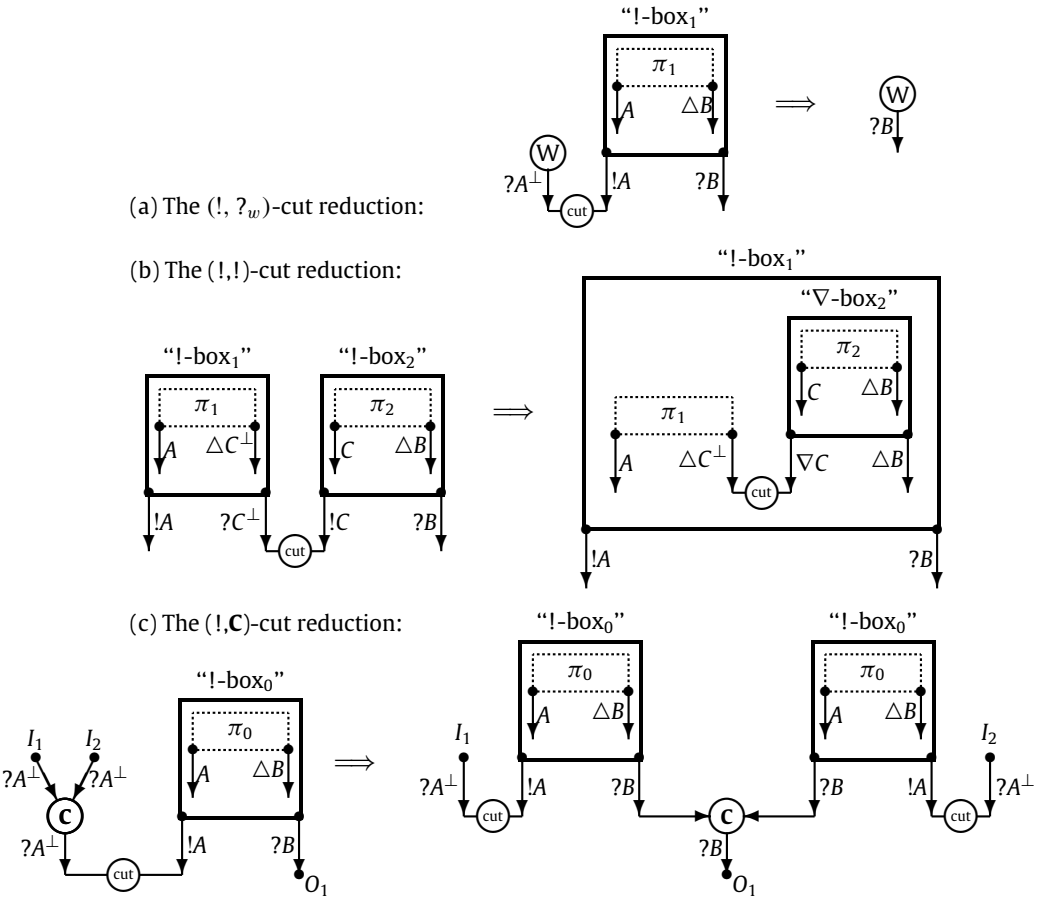


Fig. 7. The cut reduction rules for !.

But the cost of adding unrestricted weakening is the *nondeterminism of cut-elimination* (even within the intuitionistic version of Light Affine Logic):

Example 8.1 (*After Lafont*)**.** On one hand, a proof of the form (here **W** represents unrestricted weakening):

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma\vdash}}{\Gamma\vdash C}\,\mathrm{W} \qquad \dfrac{\dfrac{\pi_2}{\Sigma\vdash \Phi}}{C,\ \Sigma\vdash\Phi}\,\mathrm{W}}{\Gamma,\ \Sigma\vdash\Phi}\,\mathrm{cut} \tag{34}$$

can be reduced to

$$\dfrac{\dfrac{\pi_1}{\Gamma\vdash}}{\Gamma,\ \Sigma\vdash\Phi}\,\mathrm{W}$$

On the other hand, the same proof (34) can be also reduced to

$$\dfrac{\dfrac{\pi_2}{\Sigma\vdash\Phi}}{\Gamma,\ \Sigma\vdash\Phi}\,\mathrm{W}\quad\square$$

Example 8.1 destroys the Church–Rosser property for affine logics (even for their intuitionistic version).

As for linear logic, the 'multiplicative' nabla modality ∇ is actually all that we need to get the direct simulation for additive connectives (see Remarks 3.2 and 5.1).

Among other things, we can use this simulation in order to obtain very natural representations of programming constructs such as conditionals (cf. Example 1.1 and Remark 7.2).

In closing, because of the Church–Rosser property, the proposed *nabla* version of light linear logic LLL (as well as the *nabla* version of elementary linear logic ELL) provides the deterministic behavior of cut elimination and thereby a proper representation of *deterministic* computation. □

The last step in clarifying the rules of the system is about Easy-LLL (see Table 5), obtained from LLL-∇ by:

- Restricting '!-rule' to the form in which !C may depend on at most a single assumption of the form $\langle D\rangle$.
- Substituting $\triangle D$ for such a $\langle D\rangle$ in the just obtained '!-rule'.

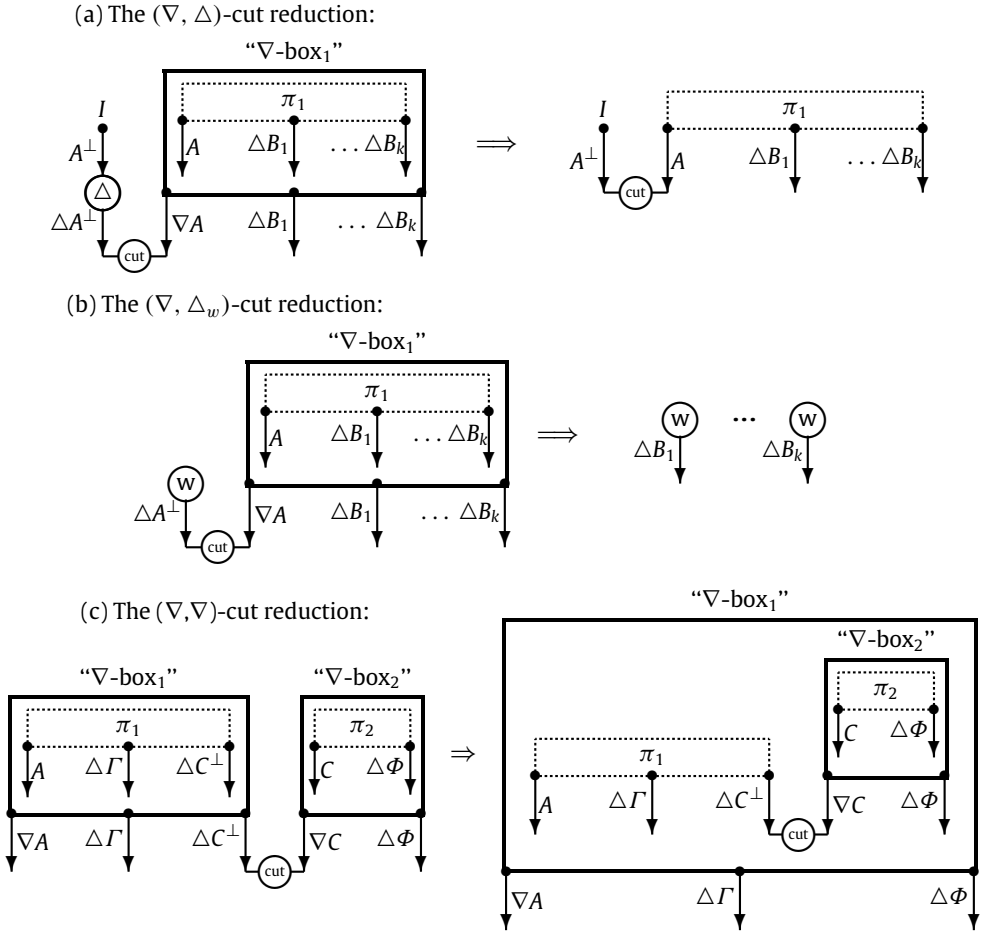


Fig. 8. The cut reduction rules for ∇.

We have proved that Easy-LLL, a traditional syntactical fragment of LLL, captures deterministic polytime computation in an 'ideal' way:

(a) The intuitionistic fragment of Easy-LLL is powerful enough to represent all polytime functions.
(b) Easy-LLL enjoys polytime strong normalization, which is within the paradigm: 'cut-elimination as a polytime computation'.
(c) Easy-LLL cut reductions enjoy the Church–Rosser property, which is within the paradigm: 'cut-elimination as a deterministic computation'.

Our proof of polytime strong normalization for Easy-LLL is based on the natural cut reductions directly produced from the rules of Easy-LLL.

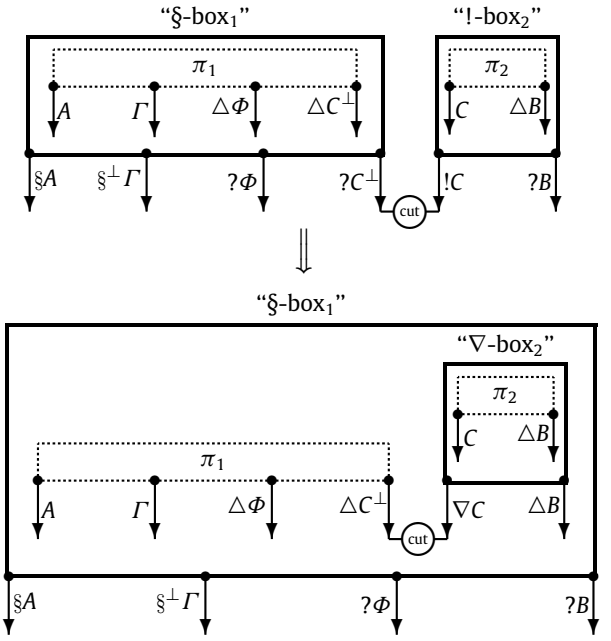
The crucial non-trivial step of the proof is that we take advantage of a restricted form of '!-rule' in Table 5 to reveal a much deeper underlying forest structure of **C**-nodes, which is stable under reductions (see Definition 6.4 and Lemma 6.2). Having detected such a stable tree-like skeleton of **C**-nodes (which represent the most problematic ?-contraction rule) provides a transparent and clean construction for polynomial bounds in Theorem 6.1.

From the point of view of the complexity of applications, we believe that Easy-LLL provides a reasonable balance between its expressive power (to capture deterministic polytime computation) and its polytime strong normalization (to guarantee polytime upper bounds):

(a) Though the syntax of Easy-LLL is traditional, Easy-LLL integrates all basic principles of Girard's light linear logic.
(b) Easy-LLL has a convincing phase semantics.
(c) Easy-LLL enjoys a transparent polytime strong normalization.
(d) Easy-LLL is flexible enough to incorporate basic constructs, such as additives/conditionals, to simulate polytime computation in a natural way (cf. [14,16,17,12]).

In closing, allowing $\ell = 2$ within '!-rule' in Table 5 yields *elementary linear logic* [8]. By the same argument we can obtain a transparent proof of strong normalization in elementary time for elementary linear logic [8].

(a) The (!,§)-cut reduction:



(b) The (§,§)-cut reduction:

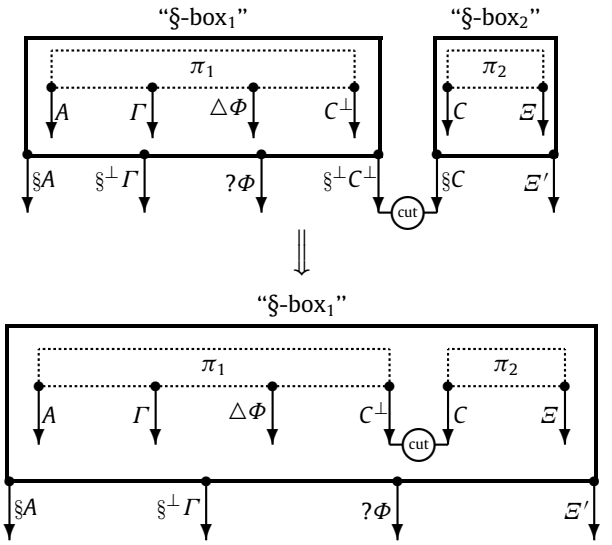


Fig. 9. The cut reduction rules for ! and §.

References

[1] A. Asperti, Light affine logic, in: Proceedings of LICS'98, 1998.
[2] A. Asperti, L. Roversi, Intuitionistic light affine logic, ACM Transactions on Computational Logic (TOCL) 3 (1) (2002) 137–175.
[3] V. Atassi, P. Baillot, K. Terui, Verification of Ptime reducibility for system F terms via Dual Light Affine Logic, in: Proceedings of CSL 2006.
[4] P. Baillot, K. Terui, Light types for polynomial time computation in lambda-calculus, in: Proceedings of LICS 2004, 2004, pp. 266–275.
[5] J.-Y. Girard, Linear logic, Theoretical Computer Science 50 (1) (1987) 1–102.
[6] J.-Y. Girard, A. Scedrov, P. Scott, Bounded linear logic: a modular approach to polynomial time computability, Theoretical Computer Science 97 (1992) 1–66.
[7] J.-Y. Girard, Linear logic: its syntax and semantics, in: J.-Y. Girard, Y. Lafont, L. Regnier (Eds.), Advances in Linear Logic, in: London Mathematical Society Lecture Notes, vol. 222, Cambridge University Press, 1995, pp. 1–42.
[8] J.-Y. Girard, Light linear logic, Information and Computation 143 (2) (1998) 175–204.
[9] Max Kanovich, Mitsuhiro Okada, Andre Scedrov, Phase semantics for light linear logic, Theoretical Computer Science 294 (3) (2003) 525–549.
[10] Max Kanovich, Light linear logic with controlled weakening, in: S. Artemov, A. Nerode (Eds.), Proc. Symposium on Logical Foundations of Computer Science, LFCS'09, Deerfield Beach, Florida, U.S.A., January 3–6, 2009, in: Lecture Notes in Computer Science, vol. 5407, 2009, pp. 246–264.
[11] Y. Lafont, Soft linear logic and polynomial time, Theoretical Computer Science 318 (2004) 163–180 (special issue on Implicit Computational Complexity).
[12] H. Mairson, K. Terui, On the computational complexity of cut-elimination in linear logic, in: Theoretical Computer Science, Proceedings of ICTCS2003, in: LNCS, vol. 2841, Springer-Verlag, 2003, pp. 23–36.

[13] D. Mazza, Linear logic and polynomial time, Mathematical Structures in Computer Science 16 (6) (2006) 947–988.
[14] A.S. Murawski, C.-H.L. Ong, On an interpretation of safe recursion in light affine logic, Theoretical Computer Science 318 (1–2) (2004) 197–223.
[15] L. Roversi, A P-time completeness proof for light logics, in: Proceedings of CSL'99, 1999, pp. 469–483.
[16] K. Terui, Light affine lambda calculus and polytime strong normalization, in: Proceedings of LICS 2001, 2001, pp. 209–220.
[17] K. Terui, Light logic and polynomial time computation, Ph.D. Thesis, Keio University, 2002.